Token Filtering: Online Attention Pruning via KV Similarity for Efficient LLM Inference

Jungmin Lee[1] **Gwanggeun Byeon**[2] **Yulhwa Kim**[3] **Seokin Hong**[2]*

[1]Department of Semiconductor and Display Engineering, Sungkyunkwan University
[2]Department of Electrical and Computer Engineering, Sungkyunkwan University
[3]Department of Semiconductor Systems Engineering, Sungkyunkwan University
{leejm518, kebyun, yulhwakim, seokin}@skku.edu

Abstract

Pruning has emerged as a promising direction for accelerating large language model (LLM) inference. However, many existing methods rely on offline calibration data, making them sensitive to distribution shifts between calibration data and inference inputs. In this paper, we introduce ***Token Filtering***, a lightweight online pruning method that selectively skips attention computation for redundant tokens during inference, without requiring any calibration data or fine-tuning. Token Filtering identifies redundant tokens based on joint key–value (KV) similarity and bypasses their attention computation. This approach reduces both compute and KV cache size while preserving essential contextual information. To preserve accuracy while meeting the target pruning ratio, we restrict pruning to later layers, which are typically less sensitive to pruning, with a layer-wise threshold that adaptively tracks the target pruning ratio. Extensive experiments on LLaMA3.1-8B, LLaMA3.1-8B-Instruct, and Qwen3-8B demonstrate that Token Filtering consistently achieves better accuracy–efficiency trade-offs than prior methods. In long-context generation, Token Filtering achieves up to 1.5× higher accuracy than the best-performing pruning baseline while reducing latency by up to 44% compared to the dense baseline.

1 Introduction

Large Language Models (LLMs) [Vaswani et al., 2017, Touvron et al., 2023, Jiang et al., 2023, Abdin et al., 2024] have demonstrated remarkable capabilities across a wide range of tasks, including natural language understanding, reasoning, and generation. However, the high inference latency and memory footprint of modern LLMs pose significant challenges for their deployment in real-world systems. In particular, attention computation dominates inference costs during decoding, motivating a growing body of research on efficient inference techniques [Zhou et al., 2024, Kwon et al., 2023].

Pruning has emerged as a promising approach for accelerating LLM inference by reducing redundant computation. Most existing pruning methods, however, are applied offline using calibration datasets to estimate the importance of model components. While effective under matched conditions, such approaches are sensitive to data mismatch and often exhibit unstable performance under input distribution shifts [Williams and Aletras, 2023, Kurz et al., 2026].

This work takes a different perspective. Instead of asking which tokens are important after attention is computed, we ask whether attention computation itself is necessary for every token during decoding. We observe that many tokens are highly redundant with respect to the past context and are unlikely to

*Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

meaningfully influence the attention outcome. This observation suggests that selectively skipping attention computation for such tokens could substantially reduce inference costs.

Motivated by this observation, we propose ***Token Filtering***, a lightweight online pruning method that makes token-level decisions before attention is executed. During inference, Token Filtering identifies redundant tokens and skips their attention computations. For such tokens, key and value representations are excluded from the KV cache. Together, these design choices reduce both attention computation and KV cache size, leading to lower inference latency and memory usage. When attention is skipped, the Transformer block preserves its functional structure through the residual connection, ensuring that pruning does not alter the model's overall behavior.

A key challenge in this setting is designing a reliable and efficient criterion for deciding whether to skip the attention computation for a token. Attention scores, which are commonly used to estimate token importance, are only available after attention computation and, therefore, cannot be used to guide attention skipping. To address this, Token Filtering leverages joint ***key–value similarity*** as a lightweight proxy for predicting a token's contribution to attention. Building on prior observations that key similarity correlates with attention scores [Park et al., 2025], we extend this insight by jointly considering both key and value representations. Tokens with key and value representations highly similar to those in the past context contribute little to the attention output and can be skipped with minimal impact on accuracy. To improve robustness under aggressive pruning, we adaptively combine key and value similarities based on their cross-head variance, avoiding erroneous pruning decisions for tokens that remain informative for some heads.

Another design question is where to apply Token Filtering across the model. Prior studies show that later Transformer layers are typically less sensitive to pruning than earlier ones [Lu et al., 2024]. Motivated by this, we propose a ***tail-focused pruning*** strategy that restricts Token Filtering to later layers. To control how aggressively each layer skips tokens, we use a similarity threshold per layer that adaptively tracks the target pruning ratio during decoding. This approach concentrates pruning on less-sensitive layers and keeps the actual pruning ratio close to the target throughout decoding.

Extensive experiments on LLaMA3.1-8B, LLaMA3.1-8B-Instruct, and Qwen3-8B demonstrate that Token Filtering achieves a strong trade-off between accuracy and efficiency across diverse pruning settings. In particular, it consistently outperforms prior pruning methods on long-context reasoning tasks, achieving up to 1.5$\times$ higher accuracy than the best-performing pruning baseline. Furthermore, Token Filtering reduces inference cost, yielding up to 44% lower latency compared to the dense baseline. These results highlight the effectiveness of Token Filtering for efficient long-context generation.

The main contributions of this paper are summarized as follows:

- We propose Token Filtering, an online pruning method that selectively skips attention computation for redundant tokens during inference, without requiring any calibration data or fine-tuning.
- We design a lightweight pruning criterion based on joint key–value similarity that adaptively combines key and value signals based on their cross-head variance, enabling robust pruning decisions even at high pruning ratios.
- We introduce a tail-focused pruning strategy with adaptive layer-wise thresholds that concentrates pruning on later, less-sensitive layers.
- Extensive experiments on LLaMA3.1-8B, LLaMA3.1-8B-Instruct, and Qwen3-8B demonstrate that Token Filtering achieves up to 1.5$\times$ higher accuracy than the best-performing pruning baseline and 44% lower latency than the dense baseline in long-context generation.

2 Related Work

Pruning. Network pruning reduces redundant computation in deep models by removing weights or larger structures such as channels, heads, or layers [Cheng et al., 2024]. For LLMs, most pruning methods operate offline using calibration datasets to estimate the importance of model components. SparseGPT [Frantar and Alistarh, 2023] addresses the layer-wise pruning problem by employing the Optimal Brain Surgeon (OBS) approach to approximate the Hessian and determine saliency. BESA [Xu et al., 2024] applies different pruning ratios per layer, adapting the sparsity allocation across the

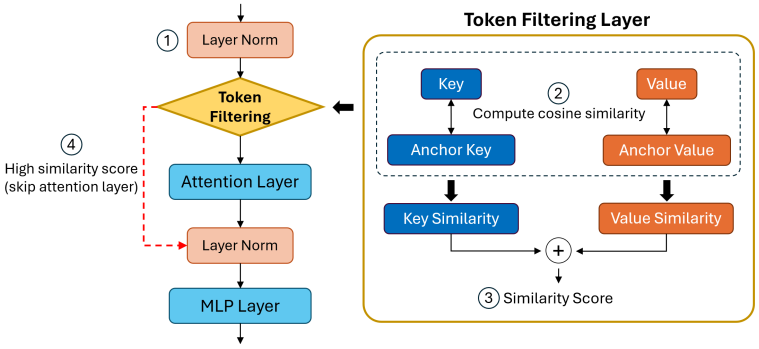


Figure 1: Overview of Token Filtering. ① Tokens pass through the Token Filtering layer before attention computation. ② The layer computes cosine similarity between the current key/value and the anchor key/value that summarize the previous context. ③ Key similarity and value similarity are combined to form a similarity score. ④ If the similarity score is high, attention computation is skipped for the token.

network. SlimGPT [Ling et al., 2024] gradually increases the pruning ratio while removing attention heads. RCPU [Haruta et al., 2025] compensates for pruning errors through rotation-constrained updates to improve structured pruning performance. Since pruning decisions in these methods are fixed before inference and do not adapt to runtime inputs, they often suffer from accuracy degradation under input distribution shifts.

More recently, Probe Pruning [Le et al., 2025] demonstrates that probing a small portion of each batch can effectively identify crucial weights, enabling online and dynamic pruning. However, it still requires a calibration dataset to construct historical states, limiting its general applicability.

KV Cache Compression. KV cache compression has been widely studied to reduce the memory and computational overhead of long-context inference. H2O [Zhang et al., 2023] prioritizes tokens based on accumulated attention scores and evicts less important ones from the KV cache. LazyLLM [Fu et al., 2024] accelerates inference by selectively computing KV states only for tokens important to the next token prediction, while lazily deferring the computation of others to later steps. HitKV [Zhao et al., 2026] compresses the KV cache by retaining tokens with high activation frequency, measured by how often they appear in the top-k attention scores during generation.

While these approaches effectively reduce KV cache size, most of them focus on memory optimization and apply token eviction after attention is computed. As a result, their impact on end-to-end decoding latency is limited. LazyLLM also skips attention computation for some tokens, but its decisions rely on attention-score-based importance estimation, which itself requires partial attention computation. In contrast, our approach makes pruning decisions before attention is executed, using a lightweight KV similarity proxy.

3 Proposed Method

This section introduces ***Token Filtering***, our method for layer-wise and token-wise online pruning in LLMs. The goal is to reduce the substantial inference cost of LLMs while maintaining accuracy at high sparsity levels. Our method has two key components: (i) a lightweight KV-similarity–based selection for real-time pruning of highly redundant tokens, where key and value similarities are adaptively combined based on their cross-head variance, and (ii) Tail-focused pruning with layer-wise thresholds that stably track the target pruning ratio across decoding. An overview of Token Filtering is illustrated in Figure 1.

3.1 KV Similarity as an Importance Proxy

As discussed above, Token Filtering performs online pruning during runtime to accelerate inference while preserving accuracy. Conventional offline pruning approaches rely on end-to-end profiling metrics collected before inference to make pruning decisions. In contrast, online pruning must make decisions on-the-fly during inference, without access to such precomputed global statistics. As a result, metrics commonly used in offline pruning, such as attention scores or cosine similarity between

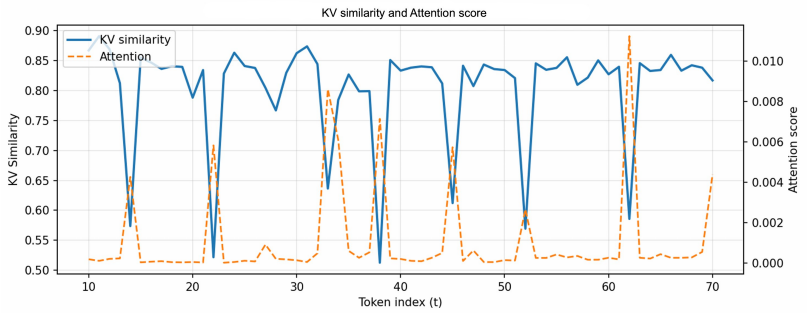


Figure 2: Visualization of KV similarity and attention scores for a representative head. We observe a clear inverse relationship, where tokens with lower KV similarity tend to receive higher attention scores.

inputs and outputs, are not available. Therefore, a new criterion is required, and we draw inspiration from input similarity to guide pruning decisions using representations available at inference time.

Prior work has examined input similarity through key representations in the attention mechanism. In particular, KeyDiff [Park et al., 2025] shows that the cosine similarity between a key and the mean of all previous keys (i.e., an anchor key) is strongly correlated with the corresponding attention score. Specifically, tokens whose keys are highly similar to the anchor key tend to receive lower attention. The underlying intuition is that key diversity is a strong proxy for global token importance. As a result, keys that are highly similar to the anchor key exhibit low diversity and capture redundant information. These keys can therefore be considered for eviction from the KV cache. However, attention pruning is more aggressive than KV cache eviction. Consequently, relying solely on key similarity to assess token importance can be unstable.

A key limitation of using key similarity alone is its sensitivity to factors unrelated to token semantics. Although keys and values are generated from the same token, keys incorporate additional information, such as positional embeddings, which can introduce variability across heads and layers. As a result, key similarity may fluctuate even when the underlying token content is largely redundant. To mitigate this issue, we incorporate value representations into the pruning criterion. Value embeddings more directly reflect token content and are less affected by positional variations. By jointly considering both key and value similarities, we obtain a more stable and reliable estimate of a token's contribution to the attention output. The detailed definition of this similarity is provided in the following subsection.

Figure 2 illustrates the relationship between key-value (KV) similarity and attention scores. The attention score is computed by accumulating attention weights across queries. We observe a clear negative correlation, where lower KV similarity corresponds to higher attention scores and higher similarity to lower scores. Lower attention scores indicate that a token receives little attention from other tokens and contributes minimally to the attention output. This observation suggests that KV similarity serves as an effective proxy for identifying tokens that make only a limited contribution to attention. Consequently, redundant attention computations can be predicted and safely skipped.

3.2 Token Filtering

By leveraging KV similarity, ***Token Filtering*** identifies redundant tokens and skips their attention computation before attention is executed. To maximize its effectiveness, we insert a Token Filtering layer before the attention sublayer in each Transformer block, as illustrated in Figure 1. The layer computes key-value similarity and selectively skips attention computation for the redundant tokens, while preserving the residual connection and preventing unnecessary KV caching. The overall procedure of the Token Filtering layer is summarized in Algorithm 1.

In the Token Filtering layer, we define KV similarity as the importance score used to determine whether to skip attention computation for a given token. KV similarity is computed as a weighted combination of key and value similarities, where the weights are determined by the relative cross-head variance of key and value similarities. Key similarity is defined as the cosine similarity between the anchor key and the current token's key. Value similarity is defined as the cosine similarity between the anchor value and the current token's value. As specified in Algorithm 1 (lines 2–3), the anchor key

Algorithm 1 Skip Decision in Token Filtering Layer

Require: layer index i, current token's keys k, current token's values v, previous keys $\mathbf{K}$, previous values $\mathbf{V}$
per-layer anchors $\textsc{AnchorK}[i], \textsc{AnchorV}[i]$, thresholds $\tau[i]$
Require: target skip ratio P_{tail}, current skip ratio $P_{current}$

1: **function** SKIPDECISION($i, k, v, \mathbf{K}, \mathbf{V}$)
2: $\text{ANCHOR}_K[i] \leftarrow \text{EMA}(\text{ANCHOR}_K[i], k)$ ▷ update key anchor
3: $\text{ANCHOR}_V[i] \leftarrow \text{EMA}(\text{ANCHOR}_V[i], v)$ ▷ update value anchor
4: $s_k \leftarrow \mu\big(\text{cosine_sim}(\textsc{AnchorK}[i], k)\big)$
5: $\sigma_k^2[i] \leftarrow \text{Var}\big(\text{cosine_sim}(\textsc{AnchorK}[i], k)\big)$
6: $s_v \leftarrow \mu\big(\text{cosine_sim}(\textsc{AnchorV}[i], v)\big)$
7: $\sigma_v^2[i] \leftarrow \text{Var}\big(\text{cosine_sim}(\textsc{AnchorV}[i], v)\big)$
8: $\alpha \leftarrow \frac{1/\sigma_k^2[i]}{1/\sigma_k^2[i]+1/\sigma_v^2[i]}$ ▷ variance-based weighting
9: $s_{\text{KV}} \leftarrow \alpha\, s_k + (1-\alpha)\, s_v$ ▷ final similarity score
10: $\tau[i] \leftarrow \tau[i] + (P_{current} - P_{tail})$ ▷ threshold adaptation
11: **if** $s_{\text{KV}} > \tau[i]$ **then**
12: Skip attention computation ▷ Do not cache k, v in KV cache
13: **end if**
14: **end function**

and value correspond to the mean of all previous keys and values. In multi-head attention, keys and values differ across heads. If we average keys and values across the head dimension and then compute cosine similarity, we can reduce computational cost. However, this approach discards head-specific information and prevents estimating variance across heads. To preserve head-level information, we compute cosine similarity for each head individually and then average the results to obtain the final key and value similarities.

Computing these similarities requires a head-specific anchor that serves as a reference for redundancy. In multi-head attention, the set of keys and values evolves with each generated token, making the exact computation of the anchor as the mean of all previous keys and values increasingly expensive during decoding. To mitigate this cost, we replace the exact average with an efficient incremental averaging strategy. Specifically, we update the anchor using a fixed smoothing factor γ:

$$\text{Anchor} \leftarrow \gamma \cdot \text{Anchor} + (1-\gamma) \cdot \mathbf{k}_{\text{current}}, \tag{1}$$

where γ is close to one (e.g., $\gamma = 0.9$). This update emphasizes recent tokens while preventing older tokens from dominating the anchor representation, effectively capturing a sliding window of recent key information. The same update rule is applied to anchor values.

Because each layer maintains only one anchor key and one anchor value (one vector per head), the memory footprint is extremely small. For reference, the total anchor size in LLaMA3.1-8B is approximately 128 KB. In contrast, the KV cache typically occupies several hundred megabytes, depending on the batch size and sequence length. When Token Filtering skips attention computation, the corresponding keys and values are not stored in the KV cache. Consequently, the memory savings achieved by KV cache pruning are orders of magnitude larger than the anchor overhead.

Since similarity is computed individually for each head, we can also measure the variance of similarity across heads (Algorithm 1, lines 4–7). In multi-head attention, each head represents a different perspective, meaning that the keys and values of different heads reflect distinct interpretations of the same token. From this viewpoint, head-wise similarity measures token redundancy across multiple perspectives. Leveraging variance helps mitigate the limitations of simple averaging. Even when the average similarity across heads is high, large variance indicates that similarity differs substantially across heads, implying that some heads still capture non-redundant and informative content.

To avoid such erroneous pruning, we downweight similarity components that exhibit high variance across heads. Consider a case where key similarity has a high average but also high variance, while value similarity has a low average and low variance. Under simple averaging, the final similarity score would fall between the two, potentially triggering pruning. In contrast, our ***variance-aware fusion***

reduces the influence of the high-variance key similarity, biasing the final score toward the value similarity with lower variance. This results in a lower overall similarity score than naive averaging and helps preserve informative tokens.

Formally, let s_k and s_v denote the cosine similarities of key and value, and let σ_k^2 and σ_v^2 be their running variances. The final similarity score is given by:

$$\alpha = \frac{\sigma_k^{-2}}{\sigma_k^{-2} + \sigma_v^{-2}}, \tag{2}$$

$$S_{\text{KV}} = \alpha\, s_k + (1 - \alpha)\, s_v. \tag{3}$$

These equations correspond to the operations in Algorithm 1, lines 8–9. Here, α is adaptively determined for each head, so that the more stable similarity component (with lower variance) contributes more to the final score. This variance-aware weighting provides a more robust criterion for attention skipping than relying solely on key similarity.

Based on this similarity score, tokens are selectively skipped before attention computation. When a token is skipped, LayerNorm is still applied as usual, and the attention output for that token is set to zero. After the residual connection, the output of the attention sublayer therefore becomes identical to its input. This design ensures that Token Filtering preserves the overall structure and functional behavior of the Transformer while skipping attention operations.

3.3 Tail-Focused Pruning with Layer-Wise Threshold

A practical concern in applying Token Filtering across all layers is that early layers are typically more sensitive to pruning, and uniform pruning can disrupt these layers and degrade accuracy. To address this, we restrict pruning to layers that are less sensitive to token removal, rather than applying it uniformly across all layers. Prior work has shown that different Transformer layers exhibit varying sensitivity to pruning. For example, [Gao et al., 2025] estimates layer-wise pruning sensitivity and applies non-uniform pruning accordingly. Similarly, [Lu et al., 2024] empirically studies layer pruning strategies and adopts a heuristic that removes the final 25% of layers.

Based on prior studies and our empirical observations, we adopt a ***tail-focused pruning*** strategy. Rather than pruning all layers uniformly, we restrict pruning to the later layers, which are generally less sensitive to token removal. Given a global pruning budget P_{target}, we select the last Y fraction of layers (e.g., $Y = 0.5$) and apply more aggressive pruning to them, such that the per-layer target pruning ratio becomes:

$$P_{\text{tail}} = \frac{P_{\text{target}}}{Y}. \tag{4}$$

Each selected layer l maintains its own learnable threshold T_l. During decoding, layer l computes a layer-wise similarity score $S_{KV}^{(l)}$ (Eq. (3)). If $S_{KV}^{(l)} > T_l$, the attention computation of that layer is skipped for the current token (Algorithm 1, lines 11–13), while the FFN is still executed. Let P_{current} denote the observed pruning ratio of layer l, defined as the fraction of decoding steps in which attention at that layer has been skipped so far. We update T_l online using proportional feedback to track the target P_{tail}:

$$T_l \leftarrow T_l + (P_{\text{current}} - P_{\text{tail}}). \tag{5}$$

Intuitively, when pruning becomes overly aggressive ($P_{\text{current}} > P_{\text{tail}}$), the threshold T_l increases, making future skips less likely; the opposite adjustment is applied when pruning is insufficient. Layers outside the tail set are never pruned. This design concentrates pruning on less sensitive layers while keeping the overall skip ratio close to the desired budget P_{target}.

To prevent unstable pruning decisions at the beginning of decoding, we introduce a short warm-up phase. During this warm-up period, the layer-wise thresholds are updated, but no pruning is performed. In all experiments, we use a short warm-up phase (i.e., 10 decoding steps), which we find sufficient to avoid cold-start instability. This ensures that pruning is triggered only after the anchor keys and values have accumulated sufficient historical information.

4 Experiments

4.1 Experimental Setup

Setup. All experiments are conducted on a single NVIDIA A100 GPU with 80GB of memory. We use pre-trained LLMs from the Hugging Face Transformers ecosystem [Wolf et al., 2020]. For long-context evaluation, we use the official LongReason [Ling et al., 2025] benchmark evaluated on the Hugging Face dataset. For baseline methods, we use the official implementations when available; otherwise, we implement them according to the descriptions in the original papers.

Models and Benchmarks. We evaluate our method on recent open-source LLMs, including LLaMA-3.1-8B [Meta AI, 2024], its instruction-tuned variant LLaMA-3.1-8B-Instruct, and Qwen3-8B [Yang et al., 2025], covering both base and instruction-tuned models across different families. We assess performance using both long-context language modeling and long-context reasoning tasks. For language modeling, we report perplexity on PG19 [Rae et al., 2019]. For long-context reasoning, we use LongReason, which tests the ability to handle extended contexts and complex reasoning over long sequences.

Implementation Details. We use a batch size of 32 for all latency experiments in Section 4.3 to evaluate throughput-oriented LLM serving scenarios. For the LongReason evaluation in Section 4.4, we reduce the batch size to 16 due to GPU memory constraints caused by long-context generation. For perplexity evaluation on PG19, where batching does not affect the results, we use a batch size of 1. The maximum generation length is set according to each task. For efficiency evaluation, we use decode lengths of 2048, 4096, and 8192 to cover both moderate and long-generation regimes, where the latency benefit of pruning becomes increasingly pronounced. For LongReason, we set the maximum generation length to 8K tokens. For perplexity evaluation on PG19, no explicit generation limit is imposed. The pruning ratio is defined as the fraction of tokens for which attention computation is skipped during decoding. All layer-wise thresholds are initialized to 0 and updated at every decoding step. We set the smoothing factor to $\gamma = 0.9$, following common practice in exponential moving averages, and fix the tail ratio to 0.5 such that pruning is applied to the latter half of the model. A warm-up phase of 10 decoding steps is applied before pruning begins. Unless otherwise specified, all hyperparameters are kept identical across models and benchmarks to ensure consistent evaluation conditions. All accuracy and latency metrics are obtained using deterministic greedy decoding. We will release the code and implementation details upon acceptance to facilitate reproducibility.

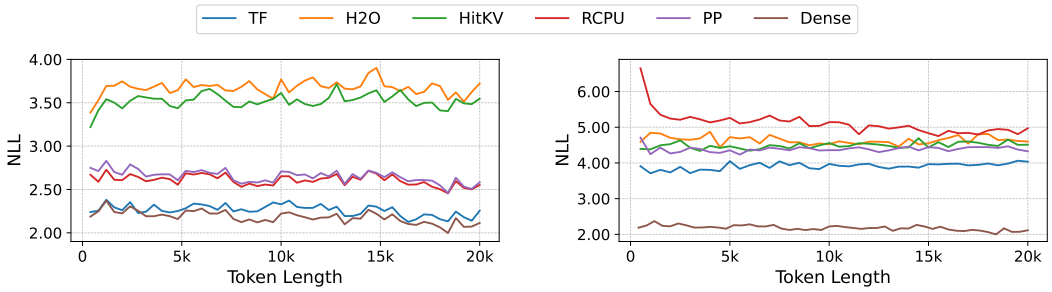
Baselines. We evaluate four representative methods from both KV cache compression and pruning-based approaches: (i) H2O [Zhang et al., 2023], a KV cache compression method that retains tokens based on cumulative attention scores. (ii) HitKV [Zhao et al., 2026], a KV cache eviction method that leverages token activation frequency via hit rates. (iii) Probe Pruning (PP) [Le et al., 2025], an online dynamic pruning method that probes a subset of tokens to guide channel pruning during inference. (iv) RCPU [Haruta et al., 2025], a structured pruning method that compensates pruning errors via rotation-constrained parameter updates.

4.2 Perplexity Results

To evaluate the impact of pruning on long-context language modeling, we measure perplexity on PG19 [Rae et al., 2019] as a function of sequence length (Figure 3).

At 20% pruning, Token Filtering remains close to the dense baseline across all sequence lengths, with an NLL gap of less than 0.2. In contrast, the next-best methods (Probe Pruning and RCPU) show a gap of around 0.4-0.6, while H2O and HitKV degrade by more than 1.0 in NLL. At 50% pruning, all methods degrade more substantially, but Token Filtering still achieves the lowest NLL of approximately 3.9, compared to 4.4 for the next-best baseline (Probe Pruning), demonstrating stronger robustness under high sparsity.

These results indicate that Token Filtering effectively preserves language modeling quality in long-context settings. Its stable performance across sequence lengths suggests that KV-similarity–based online pruning can reliably retain informative tokens, enabling robust generation even under aggressive pruning.



(a) LLaMA3.1-8B (20%) (b) LLaMA3.1-8B (50%)

Figure 3: Negative log-likelihood (NLL) as a function of sequence length on PG19.

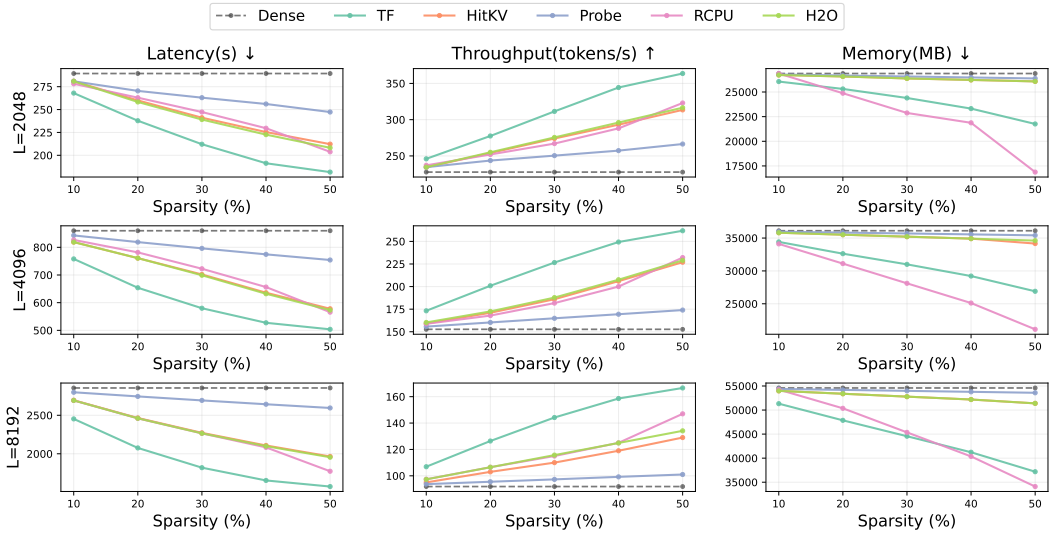


Figure 4: Latency, throughput, and memory across sparsity levels (10–50%) and decode lengths (2048, 4096, 8192) on LLaMA-3.1-8B.

4.3 Efficiency Evaluation

Figure 4 compares the efficiency of Token Filtering and baselines across sparsity levels and decode lengths on LLaMA-3.1-8B. TF consistently achieves the largest improvements in both latency and throughput, followed by RCPU, while HitKV and H2O show moderate gains, and PP performs the worst.

For memory consumption, RCPU achieves the greatest reduction because it directly reduces the model size through structured pruning, while TF achieves the second-largest reduction among the compared methods by avoiding KV cache storage for skipped tokens. Notably, the benefits of TF become more pronounced as the decode length increases. At 50% sparsity, TF improves throughput by 1.59×, 1.71×, and 1.81× at decode lengths of 2048, 4096, and 8192, respectively, while reducing latency by approximately 38%, 42%, and 45%. These results highlight the effectiveness of TF in long-generation settings.

4.4 Long-Context Reasoning Results

Table 1 compares pruning methods on LongReason. TF consistently outperforms the baselines, achieving 0.61/0.61 on LLaMA3.1-8B-Instruct and 0.62/0.54 on Qwen3-8B at 20%/25% pruning, respectively, with up to 1.5× higher accuracy than RCPU. While these results use a fixed tail ratio of $Y = 0.5$, increasing Y to 0.7 improves Qwen3-8B accuracy to 0.68/0.64, indicating model-dependent sensitivity to the tail ratio.

Table 1: LongReason accuracy results on LLaMA3.1-8B-Instruct and Qwen3-8B.

	LLaMA3.1-8B-Instruct		Qwen3-8B	
Method	20%	25%	20%	25%
Dense	0.72	-	0.78	-
TF	**0.61**	**0.61**	**0.62**	**0.54**
H2O	0.19	0.17	0.36	0.28
HitKV	0.38	0.31	0.48	0.41
Probe Pruning	0.37	0.31	0.48	0.41
RCPU	0.39	0.32	0.48	0.40

Table 2: Latency, throughput, and memory at 25% pruning on LongReason using LLaMA3.1-8B-Instruct.

Method	Latency (s)	Throughput (tokens/s)	Memory (GB)
Dense	1948.44	35.79	61.8
TF	**1087.23**	**64.13**	50.1
H2O	1169.06	59.41	55.1
HitKV	1167.32	59.42	55.1
Probe Pruning	1363.87	50.82	55.8
RCPU	1120.12	61.78	**48.2**

Table 2 shows efficiency at 25% pruning on LongReason using LLaMA3.1-8B-Instruct. TF achieves the lowest latency (1087.23 s) and highest throughput (64.13 tokens/s), while ranking second in memory usage. Compared to dense inference, TF reduces latency by approximately 44% and improves throughput by nearly 1.79×. Consistent with our findings in Section 4.3, the efficiency gains from pruning become more pronounced as the context length increases. In LongReason, where both prefilling and decoding are long, even moderate pruning leads to substantial improvements, indicating that TF becomes increasingly effective in long-context settings.

4.5 Extended Analysis

4.5.1 Pruning Focus: Tail, Head, and Uniform

We evaluate different token selection strategies by applying 20% pruning using tail-focused, head-focused, and uniform schemes, and measure the resulting perplexity (PPL). As shown in Table 3, the tail-focused strategy consistently achieves the lowest PPL across all models, followed by the uniform strategy, while the head-focused approach results in the worst performance. These results suggest that preserving tokens in the early layers is critical for maintaining model performance. Information removed in earlier layers propagates through all subsequent layers and accumulates across the network.

4.5.2 Similarity: Key, Value, and KV

Table 4 compares key, value, and joint key–value (KV) similarity for estimating token importance. To quantify alignment with attention-based importance, we report Spearman correlation between similarity-based and attention-based rankings. KV similarity achieves the strongest correlation for LLaMA models, while value similarity performs best for Qwen3-8B, with KV remaining competitive. We further evaluate agreement using F1-score by splitting tokens into top and bottom 50% groups. KV similarity consistently yields the highest F1-scores across all models, indicating better alignment with attention. In terms of downstream performance, KV similarity achieves the lowest perplexity across all models, demonstrating its effectiveness for pruning. Importantly, this improvement incurs only a marginal latency overhead (approximately 3%) compared to key-only or value-only similarity, highlighting a favorable accuracy–efficiency trade-off.

4.5.3 Scalability to Larger Models

To evaluate the scalability of Token Filtering beyond 8B-scale models, we additionally evaluate Qwen3-8B and Qwen3-32B on MMLU-Pro and LongReason under 20% pruning. For these additional experiments, we set the tail ratio to $Y = 0.7$ for both models, while the main experiments use $Y = 0.5$.

Table 3: Perplexity comparison under different token selection strategies with 20% pruning.

Model	Tail-focused	Head-focused	Uniform
LLaMA3.1-8B	**19.16**	143.8	58.11
LLaMA3.1-8B-Instruct	**22.75**	190.56	72.33
Qwen3-8B	**40.57**	168.17	74.62

Table 4: Correlation and performance metrics for different attention signals. Spearman correlation and F1-score are measured on the dense model, while PPL is evaluated under 20% pruning.

Metric	Model	Key sim.	Value sim.	KV sim.
Spearman ↓	LLaMA3.1-8B	-0.528	-0.490	**-0.543**
	LLaMA3.1-8B-Instruct	-0.607	-0.571	**-0.618**
	Qwen3-8B	-0.266	**-0.454**	-0.412
F1-score ↑	LLaMA3.1-8B	0.701	0.698	**0.714**
	LLaMA3.1-8B-Instruct	0.737	0.702	**0.739**
	Qwen3-8B	0.601	0.653	**0.663**
PPL ↓	LLaMA3.1-8B	24.99	20.08	**19.16**
	LLaMA3.1-8B-Instruct	29.68	24.60	**22.75**
	Qwen3-8B	44.32	40.72	**40.57**

Table 5: Accuracy comparison of Qwen3-8B and Qwen3-32B on MMLU-Pro and LongReason under 20% pruning with a tail ratio of $Y = 0.7$. Accuracy is reported on a 0–1 scale, and Drop denotes the absolute difference from dense inference.

Benchmark	Model	Dense ↑	TF ↑	Drop ↓
MMLU-Pro	Qwen3-8B	0.4143	0.3500	0.0643
	Qwen3-32B	0.5429	0.4821	**0.0608**
LongReason	Qwen3-8B	0.78	0.68	0.10
	Qwen3-32B	0.80	0.74	**0.06**

As shown in Table 5, Qwen3-32B achieves 48.21% accuracy on MMLU-Pro, compared to 54.29% under dense inference, corresponding to a decrease of 6.08 percentage points. This is comparable to the 6.43-point decrease observed for Qwen3-8B. On LongReason, Qwen3-32B achieves 74% accuracy compared to 80% under dense inference, resulting in a 6-point decrease, smaller than the 10-point decrease observed for Qwen3-8B.

These results suggest that pruning-induced accuracy degradation does not necessarily increase with model size, supporting the applicability of Token Filtering to larger dense models across both short-form and long-context reasoning tasks.

5 Conclusion

In this work, we propose Token Filtering, a fully online pruning technique for large language models. To address the challenges of online pruning, we leverage joint key value similarity as a lightweight importance criterion. Extensive experiments demonstrate that Token Filtering effectively reduces inference latency while maintaining strong model performance across diverse models and pruning ratios. Our method achieves these improvements without requiring calibration data or fine-tuning. In particular, Token Filtering achieves substantial latency reductions in long-context generation settings while remaining robust even at high sparsity levels. These results highlight the effectiveness of online pruning for efficient long sequence inference.

Acknowledgments and Disclosure of Funding

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.RS-2026-25521762, 40%) and Institute of Information & Communications Technology Planning & Evaluation(IITP)-ITRC(Information Technology Research Center) grant funded by the Korea government(MSIT) (IITP-2026-RS-2021-II212052, 30%) and National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.RS-2026-25490403, 30%)

References

Marah Abdin et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning (ICML)*, pages 10323–10337. PMLR, 2023.

Qichen Fu et al. Lazyllm: Dynamic token pruning for efficient long context llm inference. *arXiv preprint arXiv:2407.14057*, 2024.

Chang Gao et al. Maximum redundancy pruning: A principle-driven layerwise sparsity allocation for llms. *arXiv preprint arXiv:2503.18377*, 2025.

Shuichiro Haruta et al. Rcpu: Rotation-constrained error compensation for structured pruning of a large language model. *arXiv preprint arXiv:2510.07782*, 2025.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Simon Kurz et al. On the limitations of language-targeted pruning: Investigating the calibration language impact in multilingual llm pruning. *Transactions of the Association for Computational Linguistics*, 14:167–192, 2026.

Woosuk Kwon et al. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Qi Le et al. Probe pruning: Accelerating llms through dynamic pruning via model-probing. *arXiv preprint arXiv:2502.15618*, 2025.

Gui Ling, Ziyang Wang, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 107112–107137, 2024.

Zhan Ling et al. Longreason: A synthetic long-context reasoning benchmark via context expansion. *arXiv preprint arXiv:2501.15089*, 2025.

Yao Lu et al. Reassessing layer pruning in llms: New insights and methods. *arXiv preprint arXiv:2411.15558*, 2024.

Meta AI. Llama-3. `https://llama.meta.com/llama3/`, 2024. Accessed: 2024-05-15.

Junyoung Park et al. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *arXiv preprint arXiv:2504.15364*, 2025.

Jack W Rae et al. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Miles Williams and Nikolaos Aletras. On the impact of calibration data in post-training quantization and pruning. *arXiv preprint arXiv:2311.09755*, 2023.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, 2020. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Peng Xu et al. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. *arXiv preprint arXiv:2402.16880*, 2024.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Atlas Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Sanle Zhao et al. Hitkv: Activation frequency knows which tokens are important. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, 2026.

Z. Zhou, X. Ning, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024.

A Hyperparameter Sensitivity

In this section, we analyze the sensitivity of Token Filtering to three hyperparameters: (1) the tail-layer ratio Y, which determines the fraction of layers eligible for pruning; (2) the anchor smoothing factor γ, which controls how quickly the key/value anchors adapt to new tokens; and (3) the number of warm-up tokens, which determines when attention pruning begins.

A.1 Tail-Ratio Sensitivity

Token Filtering adopts a tail-focused pruning strategy that restricts pruning to later layers, which are generally less sensitive to token removal. Given a global pruning budget P_{target}, we select the last Y fraction of layers and distribute the pruning budget across these layers. The per-layer target pruning ratio is defined as

$$P_{\text{tail}} = \frac{P_{\text{target}}}{Y}.$$

Since the per-layer pruning ratio cannot exceed 100%, the minimum feasible value of Y depends on the global pruning budget. For example, when $P_{\text{target}} = 0.5$, the minimum valid choice is $Y = 0.5$.

Sensitivity on LLaMA3.1-8B. Table 6 reports the perplexity (PPL) of LLaMA3.1-8B under different tail-layer ratios at 20% and 50% pruning. We evaluate Y from 0.5 to 1.0, where $Y = 1.0$ corresponds to uniform pruning across all layers.

Table 6: Perplexity (PPL) of LLaMA3.1-8B under different tail-layer ratios Y. Lower is better.

Prune%	Y	PPL
20%	0.5	19.16
	0.6	21.04
	0.7	29.68
	0.8	29.22
	0.9	30.62
	1.0	58.11
50%	0.5	79.43
	0.6	123.03
	0.7	162.99
	0.8	179.02
	0.9	179.02
	1.0	416.23

As shown in Table 6, $Y = 0.5$ achieves the lowest PPL at both pruning ratios. Performance generally degrades as Y increases toward uniform pruning, with particularly severe degradation at 50% pruning. These results indicate that preserving attention computation in early layers is important for maintaining model performance.

Sensitivity on Qwen3-8B. The optimal tail-layer ratio can vary across model architectures. We additionally evaluate $Y = 0.7$ on Qwen3-8B using LongReason to examine whether distributing the same global pruning budget across more layers improves robustness.

Table 7: Effect of the tail-layer ratio Y on Qwen3-8B LongReason accuracy. The relative drop is measured from 20% to 25% pruning. Higher accuracy is better.

Y	20%	25%	Relative Drop
0.5	0.62	0.54	-12.9%
0.7	0.68	0.64	-5.9%

Table 7 shows that increasing Y from 0.5 to 0.7 improves accuracy from 0.62 to 0.68 at 20% pruning and from 0.54 to 0.64 at 25% pruning. The relative accuracy drop from 20% to 25% pruning decreases

from 12.9% to 5.9%. Unlike LLaMA3.1-8B, Qwen3-8B benefits from distributing pruning across a larger tail-layer region. These results suggest that the optimal tail ratio is architecture-dependent and that model-specific selection of Y can improve pruning robustness.

A.2 Anchor Smoothing Factor

Token Filtering maintains key and value anchors using an exponential moving average (EMA), controlled by the smoothing factor γ. A larger γ retains more historical information, while a smaller γ allows the anchors to adapt more rapidly to recent tokens.

Table 8 evaluates the effect of γ on LLaMA3.1-8B under 20% and 33% pruning.

Table 8: Perplexity (PPL) of LLaMA3.1-8B under different anchor smoothing factors γ. Lower is better.

Prune%	γ	PPL
20%	0.95	19.16
	0.90	19.16
	0.80	19.16
	0.70	18.57
	0.60	18.57
	0.50	18.57
33%	0.95	27.02
	0.90	27.02
	0.80	27.88
	0.70	27.88
	0.60	27.88
	0.50	29.22

As shown in Table 8, the effect of γ depends on the pruning ratio. At 20% pruning, smaller values of γ slightly improve PPL. However, at 33% pruning, larger values achieve better performance, suggesting that retaining historical information becomes more important under aggressive pruning. Overall, the results show relatively limited sensitivity to γ within the tested range. We therefore use $\gamma = 0.9$ as a stable default across the main experiments.

A.3 Warm-up Sensitivity

To avoid unstable pruning decisions at the beginning of decoding, Token Filtering employs a short warm-up phase during which attention remains dense while the running key and value anchors are initialized. We evaluate warm-up lengths of 0, 5, 10, 20, and 30 tokens on Qwen3-8B under 20% pruning.

Table 9: Warm-up sensitivity on Qwen3-8B under 20% pruning. Lower PPL is better.

Warm-up Tokens	PPL ↓
0	20.42
5	19.55
10	**19.21**
20	19.67
30	19.54

Table 9 shows that a short warm-up improves performance compared to immediate pruning. Without warm-up, PPL reaches 20.42, whereas 10 warm-up tokens achieve the lowest PPL of 19.21. Differences among nonzero warm-up settings are relatively small, indicating that performance is not highly sensitive to warm-up length once the anchors have been initialized. Based on these results, we use 10 warm-up tokens as the default setting throughout the main experiments.

B Overhead Analysis

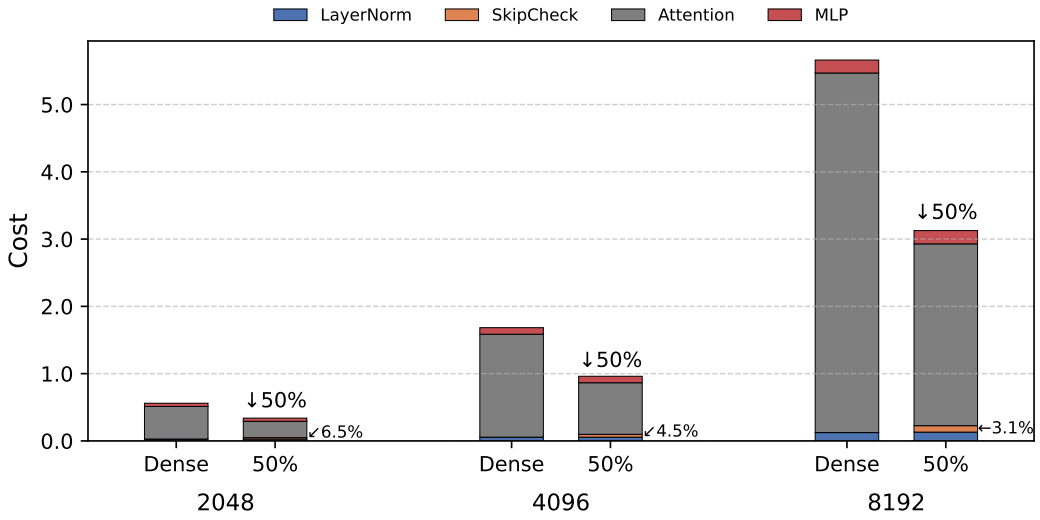


Figure 5: Breakdown of computation cost across different decode lengths and pruning settings. Attention dominates the overall computation cost, and pruning reduces the attention computation by approximately 50% across all settings. The SkipCheck overhead remains small, accounting for only 6.5%, 4.5%, and 3.1% of the total computation cost at decode lengths of 2048, 4096, and 8192, respectively. LayerNorm denotes the combined cost of the input LayerNorm and post-attention LayerNorm operations.

We further analyze the computational overhead introduced by SkipCheck. As shown in Figure 5, the additional overhead remains marginal across all settings, accounting for only 6.5%, 4.5%, and 3.1% of the total computation cost at decode lengths of 2048, 4096, and 8192, respectively. Moreover, the relative overhead decreases as the decode length increases, since the attention computation increasingly dominates the overall cost.

C Layer-wise Sensitivity and Variance-aware KV Fusion

To examine where value similarity is beneficial and to validate tail-focused pruning, we apply 40% pruning to one Transformer layer at a time and measure the relative increase in perplexity (PPL). We compare key-only similarity, an equal-weight average of key and value similarities (Equal KV), and the proposed variance-aware KV fusion. The reported values aggregate layers into regions for LLaMA3.1-8B and Qwen3-8B.

Table 10: Layer-wise sensitivity of LLaMA3.1-8B under 40% single-layer pruning. Entries show relative PPL increase (%); lower is better.

Region	Key-only	Equal KV	Variance-aware KV
L1–L15	2.11	2.49	2.24
L16–L23	0.95	1.07	0.96
L24–L30	0.56	0.58	**0.53**
L31	9.90	**9.49**	10.01

Tables 10 and 11 show strong layer dependence. For LLaMA3.1-8B, the middle-to-late pruning region (L16–L30) is relatively robust compared with the early region (L1–L15), supporting the tail-focused design. The final layer (L31), however, is notably sensitive, indicating that robustness is not strictly monotonic with depth. For Qwen3-8B, the first layer (L0) is particularly sensitive, while later regions exhibit much smaller PPL changes. The variance-aware combination performs best in LLaMA3.1-8B's L24–L30 region and Qwen3-8B's L26–L35 region, but it does not dominate

Table 11: Layer-wise sensitivity of Qwen3-8B under 40% single-layer pruning. Entries show relative PPL increase (%); lower is better.

Region	Key-only	Equal KV	Variance-aware KV
L0	66.85	**44.48**	65.86
L1–L17	**0.55**	0.75	0.65
L18–L25	**0.21**	0.25	**0.21**
L26–L35	0.38	0.38	**0.36**

Table 12: Perplexity of different anchor strategies in non-stationary and stationary PG19 settings. Lower is better.

Anchor	Non-stationary PPL	Stationary PPL
Cumulative mean	28.06	35.87
EMA (ours)	**26.35**	**34.74**

every region. These findings support selective pruning and show that the benefit of value similarity is architecture- and layer-dependent.

D Robustness of EMA Anchors under Distribution Shifts

The exponential moving average (EMA) anchor is designed to reflect recent context rather than summarize all preceding tokens equally. To test whether this recency weighting helps under distribution shifts, we evaluate two PG19 settings: a non-stationary sequence constructed by concatenating segments from different documents, creating explicit shifts at document boundaries, and a stationary sequence obtained from a continuous single document. We compare the EMA anchor against a cumulative-mean anchor that retains the full history with equal weighting.

As shown in Table 12, the EMA anchor achieves lower PPL in both settings. Relative to the cumulative mean, its PPL reduction is approximately 6.1% in the non-stationary setting and 3.2% in the stationary setting. The larger improvement under distribution shifts is consistent with the intended role of the EMA anchor as a recent-context reference.

We further analyze the first 512 tokens after a document boundary in Table 13. During the first 32 tokens, the EMA anchor is worse than the cumulative mean (68.01 vs. 64.87 PPL), showing that adaptation is not instantaneous. From tokens 33–128 onward, however, EMA improves upon the cumulative mean. Over the full 1–512 interval, EMA reduces PPL from 25.91 to 24.27, an improvement of approximately 6.3%. Thus, recency weighting improves adaptation after a short transition period, although a gap from dense inference remains.

E Direct Comparison with KeyDiff and LazyLLM

We additionally compare Token Filtering with KeyDiff and LazyLLM under a common 20% pruning budget using LLaMA3.1-8B-Instruct on 16 LongBench tasks. For this comparison, KeyDiff is adapted to a decode-time attention-skipping setting. Table 14 reports the mean accuracy and its difference from dense inference.

All three methods achieve nearly identical average accuracy under this setting, with differences of at most 0.02 percentage points among the pruned variants. We therefore do not interpret this comparison as evidence of a meaningful accuracy advantage. Instead, the methods differ primarily in how they reduce inference cost. KeyDiff was originally designed to evict KV-cache entries, reducing the cached context accessed by attention. LazyLLM selectively defers computation based on importance estimates involving attention, whereas Token Filtering makes its skip decision from key/value similarity before executing the attention sublayer. The accuracy results establish comparability under a common pruning budget; they do not, by themselves, establish a latency advantage over these two implementations.

Table 13: Post-shift perplexity by token position after a PG19 document boundary. Overall aggregates positions 1–512. Lower is better.

Anchor	1–32	33–128	129–512	Overall
Dense	**63.90**	**19.05**	**15.71**	**17.78**
Cumulative mean	64.87	26.80	23.79	25.91
EMA (ours)	68.01	26.27	21.83	24.27

Table 14: Comparison on 16 LongBench tasks with LLaMA3.1-8B-Instruct at 20% pruning. Accuracy is reported in percentage points.

Method	Accuracy ↑	Δ from Dense
Dense	41.46	–
KeyDiff	40.90	−0.56
LazyLLM	40.91	−0.55
Token Filtering (ours)	**40.92**	−0.54

Table 15: Hugging Face and vLLM Transformers-backend comparison. Lower PPL is better. The actual skip ratio is measured during decoding.

Model	Backend	Dense PPL	TF PPL (20%)	Actual Skip
LLaMA3.1-8B	Hugging Face	7.934	14.234	20.33%
LLaMA3.1-8B	vLLM	7.938	14.451	20.33%
Qwen3-8B	Hugging Face	18.703	26.558	19.62%
Qwen3-8B	vLLM	18.724	26.573	19.62%

F Compatibility with vLLM

To examine compatibility with an LLM serving framework, we implement a functional Token Filtering prototype using vLLM's Transformers backend and compare its perplexity against the Hugging Face implementation. We use the same token manifest and BF16 configuration across backends, and report results for dense inference and 20% pruning.

Table 15 shows comparable PPL across the two backends. Without pruning, the PPL gaps are 0.004 for LLaMA3.1-8B and 0.021 for Qwen3-8B. Under 20% pruning, the gaps are 0.217 and 0.015, respectively, while the measured skip ratios are identical for each model across backends. These results demonstrate functional compatibility without modifying the Token Filtering algorithm. This experiment does not measure vLLM serving throughput or latency, and we do not use it to claim an additional serving-speed improvement.

G Broader Impact

This work focuses on improving the inference efficiency of large language models through token pruning during decoding. By reducing computational overhead and memory usage, the proposed method may help lower energy consumption and improve the accessibility of deploying large language models in resource-constrained environments.

At the same time, the method is designed as a general inference optimization technique and does not directly address broader risks associated with large language models, such as misinformation, harmful content generation, or misuse in downstream applications. Since the proposed approach can also accelerate existing language models, it may indirectly facilitate both beneficial and harmful applications of such systems. We therefore encourage future work to consider efficiency improvements together with responsible deployment and safety considerations.

H Limitations

While Token Filtering demonstrates strong performance in long-context settings, several limitations remain. First, the benefits of pruning are less pronounced for short context lengths, where token redundancy is limited. Second, our method relies on key–value similarity as a proxy for token importance, which may not fully capture complex semantic dependencies, potentially leading to suboptimal decisions in certain cases. Finally, although we evaluate our approach across multiple models and benchmarks, further validation on larger-scale models and more diverse tasks is needed to fully establish its generality.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the proposed online pruning method, the use of KV similarity for token selection, and the claimed accuracy–efficiency improvements. These claims are supported by the experimental results presented in Sections 4.2–4.5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Appendix H ("Limitations") discussing reduced benefits for short contexts, the limitations of KV similarity as a proxy for token importance, and the need for broader validation on larger models and tasks.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical guarantees, theorems, or proofs. The work is primarily empirical and algorithmic.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides detailed descriptions of the experimental setup, including models, datasets, pruning settings, decoding strategies, hyperparameters, hardware configuration, and evaluation protocols necessary to reproduce the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: An anonymized implementation and reproduction scripts are included in the supplementary material for the review process. The full codebase will be publicly released upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the datasets, models, decoding configurations, pruning ratios, warm-up settings, smoothing factors, hardware, and evaluation metrics used in all experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper does not report error bars, confidence intervals, or repeated-run statistics, since all experiments are conducted using deterministic greedy decoding and therefore produce deterministic outputs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the GPU type and memory capacity used for experiments and includes latency, throughput, and memory measurements across multiple settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics. The work focuses on improving inference efficiency for existing open-source language models and does not involve human subjects or sensitive personal data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper includes a broader impact discussion in Appendix G. While the proposed method can reduce computational cost and improve the accessibility of LLM inference, potential misuse and broader societal implications are also briefly discussed.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new foundation model, generative system, or scraped dataset that would require additional safeguards for responsible release.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper properly cites the existing models, datasets, and prior methods used in the experiments, including LLaMA, Qwen, PG19, and LongReason.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release new datasets, models, or other public assets at submission time.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects research or data collection requiring IRB approval.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The work studies inference optimization for existing LLMs, but LLMs are not used as non-standard tools for conducting the research itself.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.